

PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS

PAS

PAS

PAS
PAS
PAS
PAS
PAS

PAS
PAS
PAS
PAS
PAS
PAS
PAS

```
PPPPPPPP      AAAAAA      SSSSSSSS  WW      WW  RRRRRRRR  IIIIII  IIIIII  NN      NN  TTTTTTTTTT
PPPPPPPP      AAAAAA      SSSSSSSS  WW      WW  RRRRRRRR  IIIIII  IIIIII  NN      NN  TTTTTTTTTT
PP      PP  AA      AA  SS      WW      WW  RR      RR  II      II      NN      NN  TT
PP      PP  AA      AA  SS      WW      WW  RR      RR  II      II      NN      NN  TT
PP      PP  AA      AA  SS      WW      WW  RR      RR  II      II      NN      NN  TT
PP      PP  AA      AA  SS      WW      WW  RR      RR  II      II      NN      NN  TT
PPPPPPPP      AA      AA  SSSSSS  WW      WW  RRRRRRRR  II      II      NN      NN  TT
PPPPPPPP      AA      AA  SSSSSS  WW      WW  RRRRRRRR  II      II      NN      NN  TT
PP      AAAAAAAAAA      SS      WW  WW  WW  RR  RR  II      II      NN      NN  TT
PP      AAAAAAAAAA      SS      WW  WW  WW  RR  RR  II      II      NN      NN  TT
PP      AA      AA  SS      WWW  WWW  RR      RR  II      II      NN      NN  TT
PP      AA      AA  SS      WWW  WWW  RR      RR  II      II      NN      NN  TT
PP      AA      AA  SSSSSSSS  WW      WW  RR      RR  IIIIII  IIIIII  NN      NN  TT
PP      AA      AA  SSSSSSSS  WW      WW  RR      RR  IIIIII  IIIIII  NN      NN  TT
                                     ....
                                     ....
                                     ....
                                     ....

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLLLL  IIIIII  SSSSSSSS
```



```
1 0001 0 MODULE PASSWRITE_INTEGER ( %TITLE 'Write a signed integer'
2 0002 0 IDENT = '1-002' ! File: PASWRIINT.B32 Edit: SBL1002
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: Pascal Language Support
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains a procedure which writes a signed integer
36 0036 1 to a textfile.
37 0037 1
38 0038 1 ENVIRONMENT: User mode - AST reentrant
39 0039 1
40 0040 1 AUTHOR: Steven B. Lionel, CREATION DATE: 1-April-1981
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 1-001 - Original. SBL 1-April-1981
45 0045 1 1-002 - Make total-width a longword. SBL 30-June-1982
46 0046 1 --
47 0047 1
```

```

: 49      0048 1 %SBTTL 'Declarations'
: 50      0049 1
: 51      0050 1 PROLOGUE DEFINITIONS:
: 52      0051 1
: 53      0052 1
: 54      0053 1 REQUIRE 'RTLIN:PASPROLOG';           ! Externals, linkages, PSECTs, structures
: 55      0117 1
: 56      0118 1
: 57      0119 1 TABLE OF CONTENTS:
: 58      0120 1
: 59      0121 1
: 60      0122 1 FORWARD ROUTINE
: 61      0123 1     PASSWRITE_INTEGER: NOVALUE,      ! Write to textfile
: 62      0124 1     PASSWRITEV_INTEGER: NOVALUE;     ! Write to string
: 63      0125 1
: 64      0126 1
: 65      0127 1 MACROS:
: 66      0128 1
: 67      0129 1     NONE
: 68      0130 1
: 69      0131 1 EQUATED SYMBOLS:
: 70      0132 1
: 71      0133 1     NONE
: 72      0134 1
: 73      0135 1 FIELDS:
: 74      0136 1
: 75      0137 1     NONE
: 76      0138 1
: 77      0139 1 OWN STORAGE:
: 78      0140 1
: 79      0141 1     NONE
: 80      0142 1

```



```

: 82      0143 1 %SBTTL 'PASSWRITE_INTEGER - Write an integer to textfile'
: 83      0144 1 GLOBAL ROUTINE PASSWRITE_INTEGER (      ! Write an integer
: 84      0145 1      PFV: REF $PASSPFV_FILE_VARIABLE,    ! File variable
: 85      0146 1      INTEGER,                            ! Value to write
: 86      0147 1      TOTAL_WIDTH: SIGNED,               ! Total field width
: 87      0148 1      ERROR                               ! Error unwind address
: 88      0149 1      ): NOVALUE =
: 89      0150 1
: 90      0151 1 ++
: 91      0152 1 FUNCTIONAL DESCRIPTION:
: 92      0153 1
: 93      0154 1     This procedure writes a signed integer to the specified textfile.
: 94      0155 1
: 95      0156 1 CALLING SEQUENCE:
: 96      0157 1
: 97      0158 1     CALL PASSWRITE_INTEGER (PFV.mr.r, INTEGER.rl.v, TOTAL_WIDTH.rl.v
: 98      0159 1     [ERROR.j.r])
: 99      0160 1
: 100     0161 1 FORMAL PARAMETERS:
: 101     0162 1
: 102     0163 1     PFV          - The Pascal File Variable (PFV) passed by reference.
: 103     0164 1     The structure of the PFV is defined in PASPFV.REQ.
: 104     0165 1
: 105     0166 1     INTEGER      - The integer to write.
: 106     0167 1
: 107     0168 1     TOTAL_WIDTH  - Total field width.
: 108     0169 1
: 109     0170 1     ERROR        - Optional. Address to unwind to if an error occurs.
: 110     0171 1
: 111     0172 1 IMPLICIT INPUTS:
: 112     0173 1
: 113     0174 1     NONE
: 114     0175 1
: 115     0176 1 IMPLICIT OUTPUTS:
: 116     0177 1
: 117     0178 1     NONE
: 118     0179 1
: 119     0180 1 ROUTINE VALUE:
: 120     0181 1
: 121     0182 1     NONE
: 122     0183 1
: 123     0184 1 SIDE EFFECTS:
: 124     0185 1
: 125     0186 1     If the file is the standard file INPUT or OUTPUT, it is implicitly opened.
: 126     0187 1
: 127     0188 1 SIGNALLED ERRORS:
: 128     0189 1
: 129     0190 1     LINTOOLON - line too long
: 130     0191 1     NEGWIDDIG - negative width or digits specification not allowed
: 131     0192 1
: 132     0193 1 --
: 133     0194 1
: 134     0195 2 BEGIN
: 135     0196 2
: 136     0197 2 LOCAL
: 137     0198 2     FCB: REF $PASSFCB_CONTROL_BLOCK,      ! File control block
: 138     0199 2     FIELD_WIDTH,                          ! Total width
```

```
139 0200 2      REMAINING_WIDTH,      ! Remaining width
140 0201 2      INT_DIGITS,          ! Number of digits
141 0202 2      STRING: VECTOR [12, BYTE], ! String for result
142 0203 2      DESCR: BLOCK [8, BYTE],   ! String descriptor
143 0204 2      PFV_ADDR: VOLATILE,       ! Enable argument
144 0205 2      UNWIND_ACT: VOLATILE,     ! Enable argument
145 0206 2      ERROR_ADDR: VOLATILE;     ! Enable argument
146 0207 2
147 0208 2      BUILTIN
148 0209 2      ACTUALCOUNT;
149 0210 2
150 0211 2      ENABLE
151 0212 2      PASS$IO_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR);      ! Enable error handler
152 0213 2
153 0214 2      !+
154 0215 2      ! Get ERROR parameter, if present.
155 0216 2      !-
156 0217 2
157 0218 2      IF ACTUALCOUNT () GEQU 4
158 0219 2      THEN
159 0220 2          ERROR_ADDR = .ERROR;      ! Set unwind address
160 0221 2
161 0222 2      PFV_ADDR = PFV [PFV$R_PFV];    ! Set PFV address
162 0223 2
163 0224 2      !+
164 0225 2      ! Validate PFV and get PFV.
165 0226 2      !-
166 0227 2
167 0228 2      PASS$VALIDATE_PFV (PFV [PFV$R_PFV]; FCB);
168 0229 2
169 0230 2      !+
170 0231 2      ! Set unwind action to unlock file.
171 0232 2      !-
172 0233 2
173 0234 2      UNWIND_ACT = PASS$K_UNWIND_UNLOCK;
174 0235 2
175 0236 2      !+
176 0237 2      ! Do common initialization.
177 0238 2      !-
178 0239 2
179 0240 2      PASS$INIT_WRITE (PFV [PFV$R_PFV], FCB [FCB$R_FCB]; FCB);
180 0241 2
181 0242 2      !+
182 0243 2      ! Check for invalid width.
183 0244 2      !-
184 0245 2
185 0246 2      IF .TOTAL_WIDTH LSS 0
186 0247 2      THEN
187 0248 2          $PASS$IO_ERROR (PASS$NEGWIDDIG,0);
188 0249 2
189 0250 2      !+
190 0251 2      ! Check for TOTAL_WIDTH exceeding remaining width.
191 0252 2      !-
192 0253 2
193 0254 2      REMAINING_WIDTH = .FCB [FCB$A_RECORD_END] - .FCB [FCB$A_RECORD_CUR];
194 0255 2      IF .TOTAL_WIDTH GTRU .REMAINING_WIDTH
195 0256 2      THEN
```



```

196 0257 2      $PASSIO_ERROR (PASS_LINTOOLON,1,(.TOTAL_WIDTH-.REMAINING_WIDTH));
197 0258 2
198 0259 2
199 0260 2      !+
200 0261 2      !- Try to convert directly into record buffer.
201 0262 2
202 0263 2      DESCR [DSCSW_LENGTH] = .TOTAL_WIDTH;
203 0264 2      DESCR [DSCSB_CLASS] = 0;
204 0265 2      DESCR [DSCSB_DTYPE] = 0;
205 0266 2      DESCR [DSCSA_POINTER] = .FCB [FCBSA_RECORD_CUR];
206 0267 2      IF NOT OTSSCVT_L_TI (INTEGER, DESCR)
207 0268 2      THEN
208 0269 2          BEGIN
209 0270 2
210 0271 2          !+
211 0272 2          !- Conversion failed. Try again with local buffer.
212 0273 2
213 0274 2
214 0275 2          !+
215 0276 2          !- Convert integer to text
216 0277 2
217 0278 2
218 0279 2      DESCR [DSCSW_LENGTH] = 12;
219 0280 2      DESCR [DSCSA_POINTER] = STRING;
220 0281 2
221 0282 2      OTSSCVT_L_TI (INTEGER, DESCR);          ! Can't fail
222 0283 2
223 0284 2      DESCR [DSCSA_POINTER] = CH$FIND NOT CH (12, STRING, %C' ');          ! Find first non-blank
224 0285 2      DESCR [DSCSW_LENGTH] = STRING [12] = .DESCR [DSCSA_POINTER];          ! New length
225 0286 2
226 0287 2
227 0288 2      !+
228 0289 2      !- Determine if we need a leading space.
229 0290 2
230 0291 2      FIELD_WIDTH = .TOTAL_WIDTH;
231 0292 2      INT_DIGITS = .DESCR [DSCSW_LENGTH];
232 0293 2      IF CH$RCHAR (.DESCR [DSCSA_POINTER]) NEQU %C'-' ! Negative?
233 0294 2      THEN
234 0295 2          BEGIN
235 0296 2              !+
236 0297 2              !- Not negative. Only if FIELD_WIDTH > INT_DIGITS do we have a leading blank
237 0298 2
238 0299 2              IF .FIELD_WIDTH GTR .INT_DIGITS
239 0300 2              THEN
240 0301 2                  BEGIN
241 0302 2                      DESCR [DSCSW_LENGTH] = .DESCR [DSCSW_LENGTH] + 1;
242 0303 2                      DESCR [DSCSA_POINTER] = .DESCR [DSCSA_POINTER] - 1;
243 0304 2                  END;
244 0305 2              END;
245 0306 2
246 0307 2      !+
247 0308 2      !- See if field will fit in record.
248 0309 2
249 0310 2
250 0311 2      FIELD_WIDTH = MAX (.FIELD_WIDTH, .DESCR [DSCSW_LENGTH]);
251 0312 2      BEGIN
252 0313 2      LOCAL

```



```

: 253      0314 4      EXTRA; ! Extra characters past end of line
: 254      0315 4      EXTRA = .FIELD_WIDTH - .REMAINING_WIDTH;
: 255      0316 4      IF .EXTRA GTR 0
: 256      0317 4      THEN
: 257      0318 4      $PASSIO_ERROR (PASS_LINTOOLON,1,.EXTRA);
: 258      0319 3      END;
: 259      0320 3
: 260      0321 3      !+
: 261      0322 3      ! Move leading blanks, if any.
: 262      0323 3      !-
: 263      0324 3
: 264      0325 3      IF .FIELD_WIDTH - .DESCR [DSC$W_LENGTH] GTR 0
: 265      0326 3      THEN
: 266      0327 3      FCB [FCB$A_RECORD_CUR] = CH$FILL (' ', .FIELD_WIDTH - .DESCR [DSC$W_LENGTH], .FCB [FCB$A_RECOR
: 267      0328 3
: 268      0329 3      !+
: 269      0330 3      ! Now move value
: 270      0331 3      !-
: 271      0332 3
: 272      0333 3      FCB [FCB$A_RECORD_CUR] = CH$MOVE (.DESCR [DSC$W_LENGTH], .DESCR [DSC$A_POINTER], .FCB [FCB$A_RECORD_
: 273      0334 3
: 274      0335 3      END
: 275      0336 3
: 276      0337 2      ELSE
: 277      0338 2
: 278      0339 3      BEGIN
: 279      0340 3
: 280      0341 3      !+
: 281      0342 3      ! Update record position
: 282      0343 3      !-
: 283      0344 3
: 284      0345 3      FCB [FCB$A_RECORD_CUR] = .FCB [FCB$A_RECORD_CUR] + .TOTAL_WIDTH;
: 285      0346 2      END;
: 286      0347 2
: 287      0348 2      !+
: 288      0349 2      ! Call WRITE epilogue routine to move the last character written to the
: 289      0350 2      ! user's buffer and to unlock the file variable.
: 290      0351 2      !-
: 291      0352 2
: 292      0353 2      PASS$END_WRITE (PFV [PFV$R_PFV], FCB [FCB$R_FCB]);
: 293      0354 2
: 294      0355 2      RETURN;
: 295      0356 2
: 296      0357 1      END;

```

! End of routine PASSWRITE_INTEGER

```

.TITLE PASSWRITE_INTEGER Write a signed integer
.IDENT \1-002\

.EXTRN PASSWRITE_INTEGER
.EXTRN PASSWRITE_INTEGER
.EXTRN PASSIO_HANDLER
.EXTRN PASS$VALIDATE_PFV
.EXTRN PASS$INIT_WRITE
.EXTRN PASS$SIGNAL, PASSK_NEGWIDDIG
.EXTRN PASSK_LINTOOLON
.EXTRN OT$SCVT_L_TI, PASS$END_WRITE

```


				.PSECT _PASS\$CODE,NOWRT, SHR, PIC,2		
				.ENTRY PASSWRITE_INTEGER, Save R2,R3,R4,R5,R6,R7,-		0144
				R8,R9,R10		
				MOVAB OTSS\$CVT_L_TI, R10		
				MOVAB PASS\$\$SIGNAL, R9		
				SUBL2 #28, SP		
				CLRL ERROR_ADDR		0195
				CLRL UNWIND_ACT		
				MOVAL 13\$, (FP)		
				CMPB (AP), #4		0218
				BLSSU 1\$		
				MOVL ERROR, ERROR_ADDR		0220
				MOVL PFV, R6		0222
				MOVL R6, PFV_ADDR		
				JSB PASS\$VALIDATE_PFV		0228
				MOVL #1, UNWIND_ACT		0234
				JSB PASS\$INIT_WRITE		0240
				MOVL TOTAL_WIDTH, R2		0246
				BGEQ 2\$		
				CLRL -(SP)		0248
				MOVZBL #PASS\$NEGWIDDIG, -(SP)		
				CALLS #2, PASS\$\$SIGNAL		
				RET		
				MOVAB -20(FCB), R8		0254
				SUBL3 (R8), -16(FCB), REMAINING_WIDTH		
				CML R2, REMAINING_WIDTH		0255
				BLEQU 3\$		
				SUBL3 REMAINING_WIDTH, R2, -(SP)		0257
				BRB 8\$		
				MOVZWL R2, DESCR		0263
				MOVL (R8), DESCR+4		0266
				PUSHAB DESCR		0267
				PUSHAB INTEGER		
				CALLS #2, OTSS\$CVT_L_TI		
				BLBC R0, 4\$		
				BRW 11\$		
				MOVW #12, DESCR		0279
				MOVAB STRING, DESCR+4		0280
				PUSHAB DESCR		0282
				PUSHAB INTEGER		
				CALLS #2, OTSS\$CVT_L_TI		
				SKPC #32, #12, STRING		0284
				BNEQ 5\$		
				CLRL R1		
				MOVL R1, DESCR+4		
				MOVAB STRING+12, R0		0285
				SUBW3 DESCR+4, R0, DESCR		
				MOVZWL DESCR, INT_DIGITS		0292
				CMPB @DESCR+4, #45		0293
				BEQL 6\$		
				CML FIELD_WIDTH, INT_DIGITS		0299
				BLEQ 6\$		
				INCW DESCR		0302
				DECL DESCR+4		0303
				MOVL FIELD_WIDTH, R0		0311

50	OC	AE	10	00	ED	000B9	CMPZV	#0, #16, DESCR, R0	:
				04	15	000BF	BLEQ	7\$	:
			50	OC	AE	3C 000C1	MOVZWL	DESCR, R0	:
			52		50	D0 000C5	MOVL	R0, FIELD_WIDTH	:
		50	52		53	C3 000C8	SUBL3	REMAINING_WIDTH, FIELD_WIDTH, EXTRA	0315
				OC	15	000CC	BLEQ	9\$	0316
				50	DD	000CE	PUSHL	EXTRA	0318
				01	DD	000D0	PUSHL	#1	:
			7E	00G	8F	9A 000D2	MOVZBL	#PASSK_LINTOOLON, -(SP)	:
			69		03	FB 000D6	CALLS	#3, PASS\$SIGNAL	:
					04	000D9	RET	:	
52	OC	AE	10	00	ED	000DA	CMPZV	#0, #16, DESCR, FIELD_WIDTH	0325
				11	18	000E0	BGEQ	10\$	:
			50	OC	AE	3C 000E2	MOVZWL	DESCR, R0	0327
			52		50	C2 000E6	SUBL2	R0, R2	:
52		20	6E	00	2C	000E9	MOVC5	#0, (SP), #32, R2, @0(R8)	:
				00	B8	000EE		:	
			68		53	D0 000F0	MOVL	R3, (R8)	:
	00	B8	10	OC	AE	28 000F3	MOVC3	DESCR, @DESCR+4, @0(R8)	0333
			68		53	D0 000FA	MOVL	R3, (R8)	:
				03	11	000FD	BRB	12\$	0267
			68		52	C0 000FF	ADDL2	R2, (R8)	0345
				00000000G	00	16 00102	JSB	PASS\$END_WRITE	0353
					04	00108	RET	0357	
				0000	00109		.WORD	Save nothing	0195
			50	08	AC	D0 0010B	MOVL	8(AP), R0	:
			50	04	A0	D0 0010F	MOVL	4(R0), R0	:
				E0	A0	9F 00113	PUSHAB	ERROR_ADDR	:
				E4	A0	9F 00116	PUSHAB	UNWIND_ACT	:
				E8	A0	9F 00119	PUSHAB	PFV_ADDR	:
				03	DD	0011C	PUSHL	#3	:
				5E	DD	0011E	PUSHL	SP	:
			7E	04	AC	7D 00120	MOVQ	4(AP), -(SP)	:
				00000000G	00	03 FB 00124	CALLS	#3, PASS\$IO_HANDLER	:
					04	0012B	RET	:	

; Routine Size: 300 bytes, Routine Base: _PASSCODE + 0000

; 297 0358 1
 ; 298 0359 1 !<BLF/PAGE>


```

300      0360 1 %SBTTL 'PASSWRITEV_INTEGER - Write integer to string'
301      0361 1 GLOBAL ROUTINE PASSWRITEV_INTEGER (
302      0362 1      MAX_LENGTH: WORD,
303      0363 1      STRING_LINE: REF VECTOR [, WORD],
304      0364 1      INTEGER,
305      0365 1      TOTAL_WIDTH: SIGNED,
306      0366 1      ERROR
307      0367 1      ) : NOVALUE =
308      0368 1
309      0369 1 ++
310      0370 1 FUNCTIONAL DESCRIPTION:
311      0371 1
312      0372 1     This procedure writes a signed integer to the specified string.
313      0373 1
314      0374 1 CALLING SEQUENCE:
315      0375 1
316      0376 1     CALL PASSWRITEV_INTEGER (MAX_LENGTH.rw.v, STRING_LINE.wvt.r,
317      0377 1     INTEGER.rl.v, TOTAL_WIDTH.rl.v [, ERROR.j.r])
318      0378 1
319      0379 1 FORMAL PARAMETERS:
320      0380 1
321      0381 1     MAX_LENGTH      - The maximum length of STRING_LINE.
322      0382 1
323      0383 1     STRING_LINE    - A varying string to which the output will be appended.
324      0384 1
325      0385 1     INTEGER        - The integer to write.
326      0386 1
327      0387 1     TOTAL_WIDTH   - The width of the field to write.
328      0388 1
329      0389 1     ERROR         - Optional. If specified, the address to unwind to
330      0390 1     in case of an error.
331      0391 1
332      0392 1 IMPLICIT INPUTS:
333      0393 1
334      0394 1     NONE
335      0395 1
336      0396 1 IMPLICIT OUTPUTS:
337      0397 1
338      0398 1     NONE
339      0399 1
340      0400 1 ROUTINE VALUE:
341      0401 1
342      0402 1     NONE
343      0403 1
344      0404 1 SIDE EFFECTS:
345      0405 1
346      0406 1     NONE
347      0407 1
348      0408 1 SIGNALLED ERRORS:
349      0409 1
350      0410 1     See PASSWRITE_INTEGER
351      0411 1
352      0412 1 --
353      0413 1
354      0414 2 BEGIN
355      0415 2
356      0416 2 LOCAL
```

```

357      0417 2      PFV: $PASSPFV_FILE_VARIABLE,      ! Pascal File Variable
358      0418 2      ARG_LIST: VECTOR [4, LONG],      ! Argument list
359      0419 2      PFV_ADDR: VOLATILE,              ! Enable argument
360      0420 2      UNWIND_ACT: VOLATILE,             ! Enable argument
361      0421 2      ERROR_ADDR: VOLATILE;             ! Enable argument
362      0422 2
363      0423 2      BUILTIN
364      0424 2      ACTUALCOUNT;                      ! Count of arguments
365      0425 2
366      0426 2      ENABLE
367      0427 2      PASS$IO_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR); ! Enable error handler
368      0428 2
369      0429 2      !+
370      0430 2      ! Get ERROR parameter, if present.
371      0431 2      !-
372      0432 2
373      0433 2      IF ACTUALCOUNT () GEQU 5
374      0434 2      THEN
375      0435 2      ERROR_ADDR = .ERROR;                ! Set unwind address
376      0436 2
377      0437 2      PFV_ADDR = PFV [PFV$R_PFV];         ! Set PFV address
378      0438 2
379      0439 2      !+
380      0440 2      ! Set up ARG_LIST.
381      0441 2      !-
382      0442 2
383      0443 2      ARG_LIST [0] = 3;                  ! Three arguments
384      0444 2      ARG_LIST [1] = PFV [PFV$R_PFV];     ! PFV address
385      0445 2      ARG_LIST [2] = .INTEGER;            ! Integer to write
386      0446 2      ARG_LIST [3] = .TOTAL_WIDTH;        ! Field width
387      0447 2
388      0448 2      !+
389      0449 2      ! Call PASS$DO_WRITEV to do the work, giving it the address of
390      0450 2      ! PASSWRITE_INTEGER to call.
391      0451 2      !-
392      0452 2
393      0453 2      PASS$DO_WRITEV (PFV [PFV$R_PFV], .MAX_LENGTH, STRING_LINE [0], ARG_LIST,
394      0454 2      PASSWRITE_INTEGER);
395      0455 2
396      0456 2      RETURN;
397      0457 2
398      0458 1      END;                                ! End of routine PASSWRITEV_INTEGER

```

```

                                .EXTRN PASS$DO_WRITEV
                                .ENTRY PASSWRITEV_INTEGER, Save R2,R3,R4,R5,R6
                                SUBL2 #40, SP
                                CLRL ERROR_ADDR
                                CLRL UNWIND_ACT
                                MOVAL 2$, (FP)
                                CMPB (AP), #5
                                BLSSU 1$
                                MOVL ERROR, ERROR_ADDR
                                MOVAB PFV, PFV_ADDR
                                MOVL #3, ARG_LIST
                                : 0361
                                :
                                : 0414
                                :
                                : 0433
                                :
                                : 0435
                                : 0437
                                : 0443

```


PASSWRITE_INTEG
1-002

Write a signed integer
PASSWRITEV_INTEGER - Write integer to string

G 2
16-Sep-1984 02:20:27
14-Sep-1984 12:52:05

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASWRIINT.B32;1

Page 11
(4)

10	AE	1C	AE	9E	00021	MOVAB	PFV, ARG_LIST+4	:	0444
14	AE	0C	AC	7D	00026	MOVQ	INTEGER, ARG_LIST+8	:	0445
	55	FEA5	CF	9E	0002B	MOVAB	PASSWRITE_INTEGER, R5	:	0453
	54	0C	AE	9E	00030	MOVAB	ARG_LIST, R4	:	
	56	1C	AE	9E	00034	MOVAB	PFV, R6	:	
	53	08	AC	00	00038	MOVL	STRING LINE, R3	:	
	52	04	AC	3C	0003C	MOVZWL	MAX_LENGTH, R2	:	
		00000000G	00	16	00040	JSB	PASS\$DO_WRITEV	:	
				04	00046	RET		:	0458
				0000	00047	.WORD	Save nothing	:	0414
	50	08	AC	00	00049	MOVL	8(AP), R0	:	
	50	04	A0	00	0004D	MOVL	4(R0), R0	:	
		D4	A0	9F	00051	PUSHAB	ERROR_ADDR	:	
		D8	A0	9F	00054	PUSHAB	UNWIND_ACT	:	
		DC	A0	9F	00057	PUSHAB	PFV_ADDR	:	
			03	DD	0005A	PUSHL	#3	:	
			5E	DD	0005C	PUSHL	SP	:	
		7E	AC	7D	0005E	MOVQ	4(AP), -(SP)	:	
	00000000G	00	03	FB	00062	CALLS	#3, PASS\$IO_HANDLER	:	
				04	00069	RET		:	

; Routine Size: 106 bytes, Routine Base: _PASS\$CODE + 012C

; 399 0459 1
; 400 0460 1 !<BLF/PAGE>

PASSWRITE_INTEG Write a signed integer
1-002 PASSWRITEV_INTEGER - Write integer to string

H 2
16-Sep-1984 02:20:27
14-Sep-1984 12:52:05

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASWRIINT.B32;1

Page 12
(5)

: 402 0461 1 END
: 403 0462 1
: 404 0463 0 ELUDOM

! End of module PASSWRITE_INTEGER

PSECT SUMMARY

Name	Bytes	Attributes
_PASSCODE	406	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	4	0	581	00:01.0
-\$255\$DUA28:[PASRTL.OBJ]PASLIB.L32;1	427	97	22	33	00:00.4

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:PASWRIINT/OBJ=OBJ\$:PASWRIINT MSRC\$:PASWRIINT/UPDATE=(ENH\$:PASWRIINT)

: 405 0464 0
: Size: 406 code + 0 data bytes
: Run Time: 00:09.6
: Elapsed Time: 00:27.1
: Lines/CPU Min: 2896
: Lexemes/CPU-Min: 17275
: Memory Used: 121 pages
: Compilation Complete

0298 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PASWIREH
LIS

PASWIREL
LIS

PASWIREG
LIS

PASWISTR
LIS

PATCH

PATDEF
MDL

PASWIREF
LIS

PATCH
MAP

SRMDEF
MDL

PASWIRFH
LIS

PASWIRAR
LIS

PASWIREL
LIS

PASWIRLS
LIS

PASWIRINT
LIS

PASWIRIOCT
LIS

PASWIRFG
LIS

BSTRUC
REQ

PASWIRFF
LIS

CHRKEY
REQ

PASWIRFD
LIS